1

A Paraperspective Factorization Method for Shape and Motion Recovery

Conrad J. Poelman and Takeo Kanade

29 October 1992
CMU-CS-92-208

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

Abstract

In this paper, we present a method for recovering both the shape of an object and its motion relative to the camera from a sequence of images of the object, using feature points tracked throughout the sequence. The method is based on the factorization method developed by Tomasi & Kanade, which achieves its accuracy by using a large number of points and images, and robustly applying a well-understood matrix computation, the singular value decomposition, to the highly redundant input data.

While the Tomasi & Kanade method was based on orthographic projection, our method uses a projection model known as paraperspective projection. Orthographic projection does not account for the apparent change in size of an object as it moves toward or away from the camera, nor the different angle from which an object is viewed as it moves parallel to the image plane. In contrast, paraperspective projection closely approximates perspective projection by modelling both of these effects. A new formulation based on this projection model allows us to apply the factorization method to a wider range of scenarios, and to recover the distance from the camera to the object. The method assumes no model of the motion or of the object's shape, and recovers the shape and motion accurately even for distant objects.

We present several experiments which illustrate the method's performance over a wide range of noise values and a wide range of distances from the object.

This research was partially supported by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C1465, ARPA Order No. 7597.

Keywords: computer vision, motion, shape, time-varying imagery, factorization method

1. Introduction

Recovering the geometry of a scene and the motion of the camera from a stream of images is an important task in a variety of applications, including navigation, robotic manipulation, and aerial cartography. While this is possible in principle, traditional methods have failed to produce reliable results in many situations [2].

Tomasi and Kanade [7][8] developed a robust and efficient method for accurately recovering the shape and motion of an object from a sequence of images using extracted feature points. Their method uses an orthographic projection model, which is described by linear equations. It achieves its accuracy and robustness by using a large number of frames and feature points, and by directly computing shape without computing the depth as an intermediate step. The method was tested on a variety of real and synthetic images, and was shown to perform well even for distant objects.

There are, however, some limitations of the method due to its use of the orthographic projection model. The model contains no notion at all of the distance from the camera to the object. As a result, image sequences containing large translations toward or away from the camera often produce deformed object shapes, as the method tries to explain the size differences in the images by creating size differences in the object. It also supplies no estimation of translation along the camera's optical axis, which limits its usefulness for certain tasks.

Fortunately, there exist several perspective approximations which capture more of the effects of perspective projection while remaining linear. Scaled orthographic projection, sometimes referred to as "weak perspective" [3], accounts for the scaling effect of an object as it moves towards and away from the camera. Paraperspective projection, first introduced by Ohta in [4] and named by Aloimonos in [1], accounts for the scaling effect as well as the different angle from which an object is viewed as it moves in a direction parallel to the image plane.

In this paper, we present a new factorization method based on the paraperspective projection model. The paraperspective factorization method takes a set of points extracted from the images and tracked from one image to the next, and computes the shape of the object and the motion of the camera. The method assumes no model of the motion or of the object's shape, and is still fast and robust with respect to noise. However, it can be applied to a wider realm of situations than the original factorization method, such as sequences containing significant depth translation or containing objects close to the camera, and can be used in applications where it is important to recover the distance to the object, such as navigation.

We begin by describing our camera and world reference frames and introduce the mathematical notation that we use. We review the original factorization method as defined in [8], presenting it in a slightly different manner in order to make its relation to the paraperspective method more apparent. We then present our paraperspective factorization method. We conclude with the results of some experiments which demonstrate the practicality of our system.

2. The Problem

In a shape-from-motion problem, we are given a sequence of F images taken from a camera that is moving relative to an object. Imagine that we locate P prominent feature points in the first image, and track these points from each image to the next, recording the coordinates (x_{fp}, y_{fp}) of each point p in each image f. Each feature point p that we track corresponds to a single world point, located at position $\mathbf{s}_p$ in some fixed world coordinate system. Each image f was taken at some specific camera orientation, which we describe by the orthonormal unit vectors $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$, where $\mathbf{k}_f$ points along the camera's line of sight, $\mathbf{i}_f$ corresponds to the camera image plane's x-axis, and $\mathbf{j}_f$ corresponds to the camera image's y-axis. There are of course only three independent parameters - roll, pitch, and yaw - to describe the camera's orientation, and the camera frame vectors can be written in terms of these three variables. Additionally, we describe the position of the camera in each frame f by the vector $\mathbf{t}_f$ pointing to the camera's focal point. This formulation is illustrated in Figure 1.

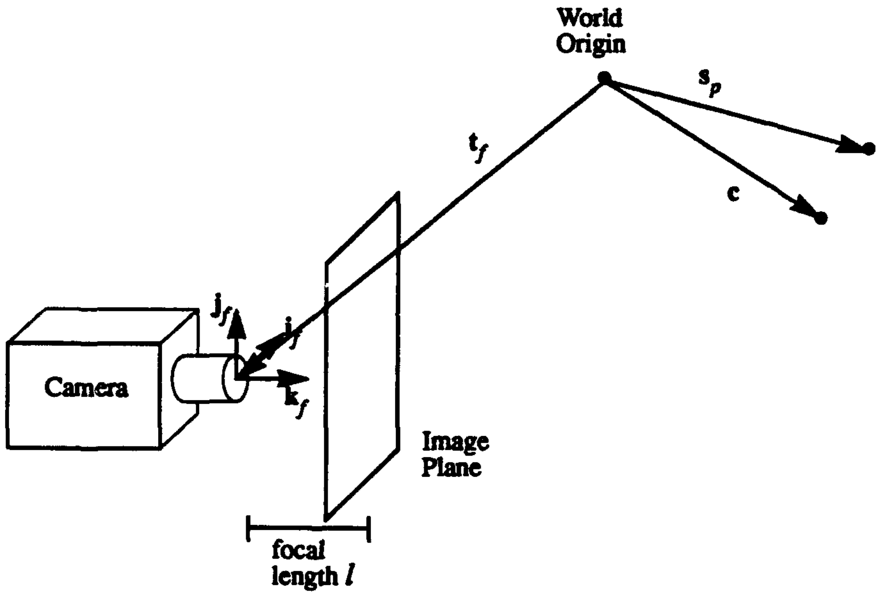

Figure 1
Coordinate System

The result of the feature tracker is set of P feature point coordinates (x_{fp}, y_{fp}) for each of the F frames of the image sequence. From this information, our goal is to recover the estimated shape of the object, given by the position $\hat{\mathbf{s}}_p$ of every point, and the estimated motion of the camera, given by $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, $\hat{\mathbf{k}}_f$ and $\hat{\mathbf{t}}_f$ for each frame in the sequence. Rather than recover $\hat{\mathbf{t}}_f$ in world coordinates, we generally recover the three separate components $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{i}}_f$, $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{j}}_f$, and $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{k}}_f$.

3. The Orthographic Factorization Method

This section presents a summary of the orthographic factorization method, which was developed by Tomasi and Kanade in 1990. A more detailed description of the method can be found in [7] or [8].

3.1. Orthographic Projection

The orthographic projection model assumes that all rays are projected from the object point parallel to the camera's line of sight so that they strike the image plane orthogonally, as illustrated in Figure 2.

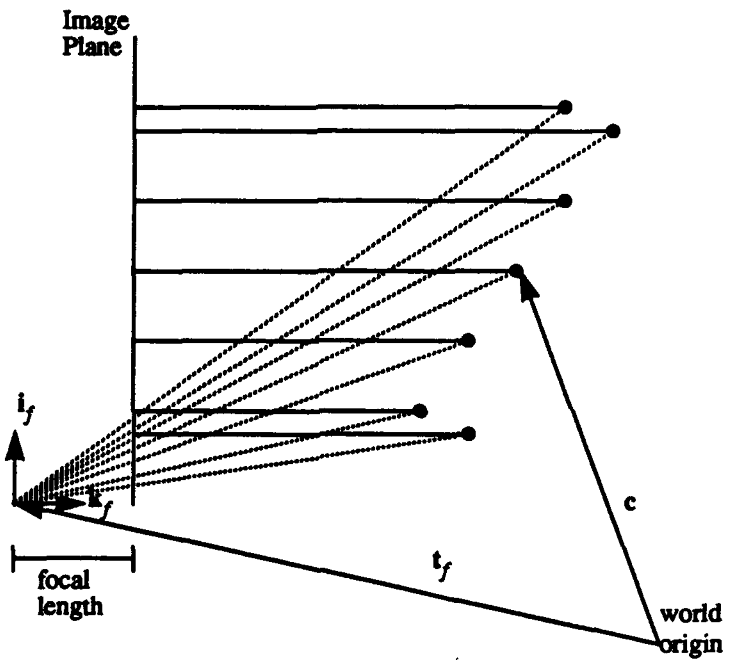

Figure 2
Orthographic Projection in two dimensions
Dotted lines indicate true perspective projection

Under orthographic projection, a point p whose location is $\mathbf{s}_p$ will be observed in frame f at image coordinates (x_{fp}, y_{fp})

$$x_{fp} = \mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) \qquad y_{fp} = \mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)\,. \tag{1}$$

We can rewrite these equations as

$$x_{fp} = \mathbf{m}_f \cdot \mathbf{s}_p + cx_f \qquad y_{fp} = \mathbf{n}_f \cdot \mathbf{s}_p + cy_f, \tag{2}$$

where

$$c x_f = -(\mathbf{t}_f \cdot \mathbf{i}_f) \qquad c y_f = -(\mathbf{t}_f \cdot \mathbf{j}_f) \tag{3}$$

$$\mathbf{m}_f = \mathbf{i}_f \qquad \mathbf{n}_f = \mathbf{j}_f. \tag{4}$$

3.2. Decomposition

We organize all of the feature point coordinates (x_{fp}, y_{fp}) into a $2F \times P$ *measurement matrix* W such that

$$W = \begin{bmatrix} x_{11} & \cdots & x_{1P} \\ \cdots & \cdots & \cdots \\ x_{F1} & \cdots & x_{FP} \\ y_{11} & \cdots & y_{1P} \\ \cdots & \cdots & \cdots \\ y_{F1} & \cdots & y_{FP} \end{bmatrix}. \tag{5}$$

Each column of the measurement matrix contains all the observations for a single point, while each row contains all the observed x-coordinates or y-coordinates for a single frame. We can combine equation (2) for all points and all frames into the single matrix equation

$$W = MS + T\begin{bmatrix} 1 & \ldots & 1 \end{bmatrix}, \tag{6}$$

where M is the $2F \times 3$ motion matrix, S is the $3 \times P$ shape matrix, and T is a $2F \times 1$ translation vector.

Up to this point we have not put any restrictions on the location of the world origin, except that it be stationary with respect to the object. For simplicity, we set the world origin at the center-of-mass of the object, denoted by $\mathbf{c}$, so that

$$\mathbf{c} = \frac{1}{P} \sum_{p=1}^{P} \mathbf{s}_p = 0. \tag{7}$$

This enables us to compute the i^{th} element of the translation vector T directly from W, simply as the average of the i^{th} row of the measurement matrix. We then subtract out the translation from W, leaving us with a "registered" matrix W^*. Because W^* is the product of a $2F \times 3$ motion matrix M and the $3 \times P$ shape matrix S, it's rank is at most 3. We use singular value decomposition to factor W^* into

$$W^* = \hat{M}\hat{S}, \tag{8}$$

where the product $\hat{M}\hat{S}$ is the best possible rank three approximation to W^*, in the sense of minimizing the sum of squares difference between corresponding elements of W^* and $\hat{M}\hat{S}$.

3.3. Normalization

The decomposition of equation (8) is not unique. In fact, any 3×3 non-singular matrix A and its inverse could be inserted between $\hat{M}$ and $\hat{S}$, and their product would still equal W'. Thus the actual motion and shape are given by

$$M = \hat{M}A \qquad S = A^{-1}\hat{S}' \tag{9}$$

when the 3×3 invertible matrix A is selected appropriately. We determine this matrix A by observing that the motion matrix M must be of a certain form. Because $\mathbf{i}_f$ and $\mathbf{j}_f$ are unit vectors, we derive from equation (4) that

$$|\mathbf{m}_f|^2 = 1 \qquad |\mathbf{n}_f|^2 = 1, \tag{10}$$

and because they are orthogonal,

$$\mathbf{m}_f \cdot \mathbf{n}_f = 0. \tag{11}$$

Equations (10) and (11) give us $3F$ equations which we call the *metric constraints*. These constraints have the property that they are linear in the 6 unique elements of the symmetric positive definite 3×3 matrix $Q = A^T A$, making the problem a simple linear data-fitting problem which can be easily solved for its least sum-of-squares error solution. Once the symmetric matrix Q has been determined, as long as it is a positive definite matrix, it is a simple matter to find A.

3.4. Shape and Motion Recovery

Once the matrix A has been found, the shape and motion can easily be computed from equation (9). In the case that the resulting motion matrix M still does not exactly satisfy the metric constraints (since we have only solved for the A which provides the best fit to the metric constraints), we simply normalize each row of M and compute the orthonormal $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$ which are closest to our computed values. There is an ambiguity in the solution due to an arbitrary world coordinate orientation. We eliminate it by aligning the world axes with the first frame's camera axes, so that $\hat{\mathbf{i}}_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ and $\hat{\mathbf{j}}_1 = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$.

3.5. Performance

The orthographic factorization method has been tested on a variety of real image sequences and has been shown to perform extremely well. In a laboratory experiment in which the camera motion was carefully controlled and recorded, the method recovered the rotation of the camera with a maximum error of less than 0.4 degrees, and with an average error far less. The object shape was also recovered accurately, with errors between the measured and computed distances between pairs of points generally less than 1 percent.

The method was also tested on an outdoor scene, in which a hand-held camera was used to view the corner of a house. Despite the obvious jerky, non-smooth motion of the camera, the method was able to recover the shape of the house and the motion of the camera [8].

4. Paraperspective Factorization Method

4.1. Paraperspective Projection

Under orthographic projection, as an object translates along the camera's optical axis, its size in the image remains constant, and as an object translates parallel to the image plane, its image simply translates as well. Under perspective projection, however, an object's size changes as it translates along the camera's optical axis; furthermore, as an object translates parallel to the image plane, the image both translates and presents a slightly different view of the object, making the object appear to rotate. The former effect is the "scaling effect", while the latter is sometimes referred to as the "position effect" of perspective projection [1], because the amount of apparent rotation depends on the object's position in the image relative to the center of projection.

In this paper, we use an approximation to perspective projection known as paraperspective projection, which was introduced by Ohta in order to solve a shape from texture problem. Paraperspective projection closely approximates perspective projection by modelling both the scaling effect and the position effect, while retaining some of the linear properties of orthographic projection. The paraperspective projection of an object onto an image, illustrated in Figure 3, involves two steps.

1. The points of the object are projected along the direction of the line connecting the focal point of the camera to the object's center-of-mass, onto a plane parallel to the image plane and passing through the object's center-of-mass.

2. These points are then projected onto the image plane using perspective projection. Because the points are all on a plane parallel to the image plane, this is equivalent to simply scaling the image by the ratio of the camera focal length and the distance along the camera's optical axis to the object's center-of-mass.[1]

In general, the projection of a point $\mathbf{p}$ along direction $\mathbf{r}$, onto the plane with normal $\mathbf{n}$ and distance from the origin d, is given by the equation $\mathbf{p}' = \mathbf{p} - \frac{\mathbf{p} \cdot \mathbf{n} - d}{\mathbf{r} \cdot \mathbf{n}} \mathbf{r}$. Figure 4 illustrates the first step of paraperspective projection. We project the object point $\mathbf{s}_p$ along the direction $\mathbf{c} - \mathbf{t}_f$, which is the direction from the camera's focal point to the object's center-of-mass, onto the plane defined by normal $\mathbf{k}_f$ and distance from the origin $\mathbf{c} \cdot \mathbf{k}_f$, giving

$$\mathbf{s}'_{fp} = \mathbf{s}_p - \frac{(\mathbf{s}_p \cdot \mathbf{k}_f) - (\mathbf{c} \cdot \mathbf{k}_f)}{(\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f} (\mathbf{c} - \mathbf{t}_f). \tag{12}$$

The perspective projection of these points onto the image plane is given by subtracting $\mathbf{t}_f$ from $\mathbf{s}'_{fp}$ to give the position of the point in the camera's coordinate system, and then scaling the result by the ratio of the camera's focal length to z_f, the depth to the object's center-of-mass, where $z_f = (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f$. This yields the coordinates of the projection in the image

1. The scaled orthographic projection model (also known as "weak perspective") is similar to paraperspective projection, except that the direction of the initial projection is parallel to the camera's optical axis rather than parallel to the line connecting the object's center-of-mass to the camera's focal point. This model captures the scaling effect of perspective projection, but not the position effect. For discussion of scaled orthographic projection, see Appendix I.

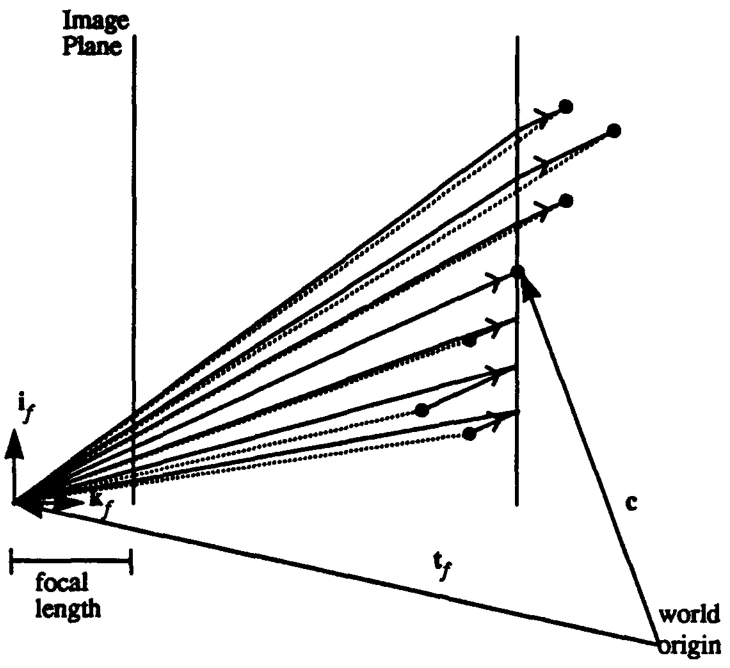

Figure 3
Paraperspective Projection in two dimensions
Dotted lines indicate true perspective projection
⟶⟶ indicate parallel lines.

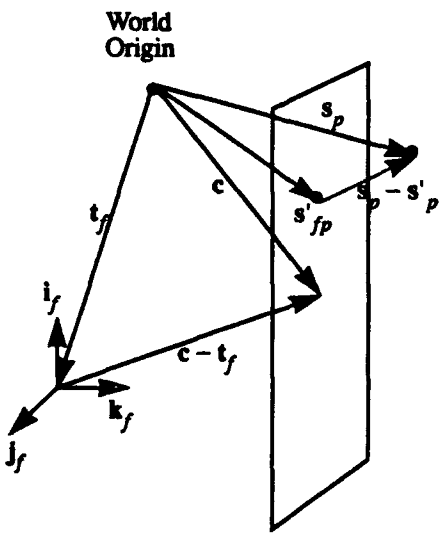

Figure 4
Projection of a point onto a plane parallel to the image plane and passing through the object's center-of-mass

plane,

$$x_{fp} = \frac{l\mathbf{i}_f}{z_f}(\mathbf{s}'_{fp} - \mathbf{t}_f) \qquad y_{fp} = \frac{l\mathbf{j}_f}{z_f}(\mathbf{s}'_{fp} - \mathbf{t}_f). \tag{13}$$

Substituting (12) into (13) and simplifying yields the general paraperspective equations for x_{fp} and y_{fp}

$$\begin{aligned} x_{fp} &= \frac{l}{z_f}\{\left[\mathbf{i}_f - \frac{\mathbf{i}_f \cdot (\mathbf{c} - \mathbf{t}_f)}{z_f}\mathbf{k}_f\right] \cdot (\mathbf{s}_p - \mathbf{c}) + (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{i}_f\} \\ y_{fp} &= \frac{l}{z_f}\{\left[\mathbf{j}_f - \frac{\mathbf{j}_f \cdot (\mathbf{c} - \mathbf{t}_f)}{z_f}\mathbf{k}_f\right] \cdot (\mathbf{s}_p - \mathbf{c}) + (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{j}_f\} \end{aligned} \tag{14}$$

For simplicity, we assume unit focal length, $l = 1$.

We have not up to this point put any requirements on our world coordinate system except that it be stationary with respect to the object. Thus we see that we can simplify our equations by placing the world origin at the object's center-of-mass, or setting $\mathbf{c} = 0$, thus reducing (14) to

$$\begin{aligned} x_{fp} &= \frac{1}{z_f}\{\left[\mathbf{i}_f + \frac{\mathbf{i}_f \cdot \mathbf{t}_f}{z_f}\mathbf{k}_f\right] \cdot \mathbf{s}_p - (\mathbf{t}_f \cdot \mathbf{i}_f)\} \\ y_{fp} &= \frac{1}{z_f}\{\left[\mathbf{j}_f + \frac{\mathbf{j}_f \cdot \mathbf{t}_f}{z_f}\mathbf{k}_f\right] \cdot \mathbf{s}_p - (\mathbf{t}_f \cdot \mathbf{j}_f)\} \end{aligned} \tag{15}$$

These equations can be rewritten as

$$x_{fp} = \mathbf{m}_f \cdot \mathbf{s}_p + cx_f \qquad y_{fp} = \mathbf{n}_f \cdot \mathbf{s}_p + cy_f \tag{16}$$

where

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{17}$$

$$cx_f = -\frac{\mathbf{t}_f \cdot \mathbf{i}_f}{z_f} \qquad cy_f = -\frac{\mathbf{t}_f \cdot \mathbf{j}_f}{z_f} \tag{18}$$

$$\mathbf{m}_f = \frac{\mathbf{i}_f - cx_f\mathbf{k}_f}{z_f} \qquad \mathbf{n}_f = \frac{\mathbf{j}_f - cy_f\mathbf{k}_f}{z_f}. \tag{19}$$

Notice that equation (16) is identical to its counterpart for orthographic projection, equation (2), although the corresponding definitions of cx_f, cy_f, $\mathbf{m}_f$, and $\mathbf{n}_f$ differ somewhat. This similarity enables us to perform the basic decomposition of the matrix in exactly the same manner as we did for orthographic projection.

4.2. Decomposition

We can combine equation (16), for all points p from 1 to P, and all frames f from 1 to F, into the single matrix equation

$$\begin{bmatrix} x_{11} & \cdots & x_{1P} \\ \cdots & \cdots & \cdots \\ x_{F1} & \cdots & x_{FP} \\ y_{11} & \cdots & y_{1P} \\ \cdots & \cdots & \cdots \\ y_{F1} & \cdots & y_{FP} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 \\ \cdots \\ \mathbf{m}_F \\ \mathbf{n}_1 \\ \cdots \\ \mathbf{n}_F \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_P \end{bmatrix} + \begin{bmatrix} cx_1 \\ \cdots \\ cx_F \\ cy_1 \\ \cdots \\ cy_F \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}, \tag{20}$$

or

$$W = MS + T\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}, \tag{21}$$

where W is the $2F \times P$ measurement matrix, M is the $2F \times 3$ motion matrix, S is the $3 \times P$ shape matrix, and T is the $2F \times 1$ translation vector. We have set $\mathbf{c} = 0$, so by definition

$$\mathbf{c} = \frac{1}{P} \sum_{p=1}^{P} \mathbf{s}_p = 0. \tag{22}$$

Using this and equation (16) we can write

$$\begin{aligned} \sum_{p=1}^{P} x_{fp} &= \sum_{p=1}^{P} (\mathbf{m}_f \cdot \mathbf{s}_p + cx_f) = \mathbf{m}_f \cdot \sum_{p=1}^{P} \mathbf{s}_p + Pcx_f = Pcx_f \\ \sum_{p=1}^{P} y_{fp} &= \sum_{p=1}^{P} (\mathbf{n}_f \cdot \mathbf{s}_p + cy_f) = \mathbf{n}_f \cdot \sum_{p=1}^{P} \mathbf{s}_p + Pcy_f = Pcy_f \end{aligned} \tag{23}$$

Therefore we can compute cx_f and cy_f immediately from the image data as

$$cx_f = \frac{1}{P} \sum_{p=1}^{P} x_{fp} \qquad cy_f = \frac{1}{P} \sum_{p=1}^{P} y_{fp}. \tag{24}$$

We then subtract these values from the corresponding rows in W, giving the registered measurement matrix

$$W^* = \begin{bmatrix} x_{11} & \cdots & x_{1P} \\ \cdots & \cdots & \cdots \\ x_{F1} & \cdots & x_{FP} \\ y_{11} & \cdots & y_{1P} \\ \cdots & \cdots & \cdots \\ y_{F1} & \cdots & y_{FP} \end{bmatrix} - \begin{bmatrix} cx_1 \\ \cdots \\ cx_F \\ cy_1 \\ \cdots \\ cy_F \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 \\ \cdots \\ \mathbf{m}_F \\ \mathbf{n}_1 \\ \cdots \\ \mathbf{n}_F \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_P \end{bmatrix}. \tag{25}$$

Since W^* is the product of two matrices each of rank at most 3, W^* has rank at most 3, just as it did in the orthographic projection case. If there is noise present, the rank of W^* will not be exactly 3, but by computing the Singular Value Decomposition (SVD) of W^* and only retaining the largest 3 singular values, we can factor W^* into

$$W^* = \hat{M}\hat{S}, \tag{26}$$

where M is a $2F \times 3$ matrix and S is a $3 \times P$ matrix. Using the SVD to perform this factorization guarantees that the product $\hat{M}\hat{S}$ is the best possible rank 3 approximation to W^*, in the sense that it minimizes the sum of squares difference between corresponding elements of W^* and $\hat{M}\hat{S}$.

4.3. Normalization

Just as in the orthographic case, the decomposition of W^* into the product of $\hat{M}$ and $\hat{S}$ is not unique. Any 3×3 matrix A and its inverse could be inserted between $\hat{M}$ and $\hat{S}$, and the product would still equal W^*. We need to find the matrix A that gives the true shape and motion

$$\begin{aligned} M &= \hat{M}A \\ S &= A^{-1}\hat{S} \end{aligned} \tag{27}$$

Again, we determine this matrix A by observing that the motion matrix M must be of a certain form. As in the orthographic case, we take advantage of the fact that $\mathbf{i}_f$ and $\mathbf{j}_f$ are unit vectors and are orthogonal. According to equation (19), we observe that

$$|\mathbf{m}_f|^2 = \frac{1 + cx_f^2}{z_f^2} \qquad |\mathbf{n}_f|^2 = \frac{1 + cy_f^2}{z_f^2}. \tag{28}$$

We know the values of cx_f and cy_f from our initial registration step, but because we do not know the value of the depth z_f, we cannot impose individual constraints on the magnitudes of $\mathbf{m}_f$ and $\mathbf{n}_f$ as we did in the orthographic factorization method. However, from equation (28) we see that

$$\frac{1}{z_f^2} = \frac{|\mathbf{m}_f|^2}{1 + cx_f^2} = \frac{|\mathbf{n}_f|^2}{1 + cy_f^2}. \tag{29}$$

Therefore we adopt the following constraint on the magnitudes of $\mathbf{m}_f$ and $\mathbf{n}_f$:

$$\frac{|\mathbf{m}_f|^2}{1 + cx_f^2} - \frac{|\mathbf{n}_f|^2}{1 + cy_f^2} = 0. \tag{30}$$

In the case of orthographic projection, one constraint on $\mathbf{m}_f$ and $\mathbf{n}_f$ was that they each have unit magnitude, as required by equation (10). In the above paraperspective version of those constraints, we simply require that their magnitudes be in a certain ratio.

There is also a constraint on the angle relationship of $\mathbf{m}_f$ and $\mathbf{n}_f$. From the definition of $\mathbf{m}_f$ and $\mathbf{n}_f$, we have

$$\mathbf{m}_f \cdot \mathbf{n}_f = \frac{cx_f cy_f}{z_f^2}. \tag{31}$$

The problem with this constraint is that, again, z_f is unknown. We could choose to use either value from equation (29) for $1/z_f^2$, since theoretically they should be equal, but in order to avoid relying on some values more than others, we use the average of the two quantities. We choose the arithmetic mean over the geometric mean or some other measure in order to keep the constraints linear in $Q = A^T A$. Thus our second constraint becomes

$$\mathbf{m}_f \cdot \mathbf{n}_f - \frac{cx_f cy_f |\mathbf{m}_f|^2}{2(1+cx_f^2)} - \frac{cx_f cy_f |\mathbf{n}_f|^2}{2(1+cy_f^2)} = 0. \tag{32}$$

This is the paraperspective version of the orthographic constraint given by equation (11), which required that the dot product of $\mathbf{m}_f$ and $\mathbf{n}_f$ be zero.

Equations (30) and (32) are homogeneous constraints, which could be trivially satisfied by the solution $M = 0$. To avoid this solution, we impose the additional constraint that

$$|\mathbf{m}_1| = 1. \tag{33}$$

This does not effect the final solution except by a scaling factor.

Equations (30), (32), and (33) are the paraperspective version of the *metric constraints*, and we compute the 3×3 matrix A such that $M = \hat{M}A$ best satisfies the metric constraints in the least sum-of-squares error sense. This is a simple problem because we have been careful to ensure that they are linear constraints in the 6 unique elements of the symmetric 3×3 matrix $Q = A^T A$. We use the metric constraints to compute Q, compute its Jacobi Transformation $Q = L \Lambda L^T$, where Λ is the diagonal eigenvalue matrix, and as long as Q is positive definite, A is

$$A = (L \Lambda^{1/2})^T. \tag{34}$$

4.4. Shape and Motion Recovery

Once the matrix A has been determined, the shape matrix S and the motion matrix M can easily be computed from equation (27). For each frame f, however, there is a more complex relationship between the actual translation and rotation vectors and the rows of the matrix M. From equation (19) we can see that

$$\hat{\mathbf{i}}_f = z_f \mathbf{m}_f + cx_f \hat{\mathbf{k}}_f \qquad \hat{\mathbf{j}}_f = z_f \mathbf{n}_f + cy_f \hat{\mathbf{k}}_f. \tag{35}$$

We use this to determine the following equations:

$$\begin{gathered} \hat{\mathbf{i}}_f \times \hat{\mathbf{j}}_f = (z_f \mathbf{m}_f + cx_f \hat{\mathbf{k}}_f) \times (z_f \mathbf{n}_f + cy_f \hat{\mathbf{k}}_f) = \hat{\mathbf{k}}_f \\ |\hat{\mathbf{i}}_f| = |z_f \mathbf{m}_f + cx_f \hat{\mathbf{k}}_f| = 1 \\ |\hat{\mathbf{j}}_f| = |z_f \mathbf{n}_f + cy_f \hat{\mathbf{k}}_f| = 1 \end{gathered} \tag{36}$$

Again, we do not know a value for z_f, but using the relations specified in equation (29) and the additional knowledge that $|\hat{\mathbf{k}}_f| = 1$, equation (36) can be reduced to

$$\begin{gathered}(\tilde{\mathbf{m}}_f \times \tilde{\mathbf{n}}_f) \cdot \hat{\mathbf{k}}_f = 1 \\ \tilde{\mathbf{m}}_f \cdot \hat{\mathbf{k}}_f = -cx_f \\ \tilde{\mathbf{n}}_f \cdot \hat{\mathbf{k}}_f = -cy_f\end{gathered} \tag{37}$$

where $\tilde{\mathbf{m}}_f = \sqrt{1 + cx_f^2}\frac{\mathbf{m}_f}{|\mathbf{m}_f|}$ and $\tilde{\mathbf{n}}_f = \sqrt{1 + cy_f^2}\frac{\mathbf{n}_f}{|\mathbf{n}_f|}$. These constraints are put in the form

$$G_f\hat{\mathbf{k}}_f = H_f, \tag{38}$$

where

$$G_f = \begin{bmatrix}(\tilde{\mathbf{m}}_f \times \tilde{\mathbf{n}}_f) \\ \tilde{\mathbf{m}}_f \\ \tilde{\mathbf{n}}_f\end{bmatrix} \qquad H_f = \begin{bmatrix}1 \\ -cx_f \\ -cy_f\end{bmatrix} \tag{39}$$

We compute $\hat{\mathbf{k}}_f$ simply as

$$\hat{\mathbf{k}}_f = G_f^{-1}H_f \tag{40}$$

and then compute

$$\hat{\mathbf{i}}_f = \tilde{\mathbf{n}}_f \times \hat{\mathbf{k}}_f \qquad \hat{\mathbf{j}}_f = \hat{\mathbf{k}}_f \times \tilde{\mathbf{m}}_f. \tag{41}$$

Because $\mathbf{m}_f$ and $\mathbf{n}_f$ may not have exactly satisfied the metric constraints, there is no guarantee that the $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$ given by this equation will be orthonormal. Therefore we actually compute the orthonormal $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$ which are closest to the values given by equation (41). Due to the arbitrary world coordinate orientation, to obtain a unique solution we then rotate the computed shape and motion to align the world axes with the first frame's camera axes, so that $\hat{\mathbf{i}}_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ and $\hat{\mathbf{j}}_1 = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$.

All that remain to be computed are the translations for each frame. We calculate the depth z_f from either part or some combination of the parts of equation (29). Since we already know cx_f, cy_f, $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$, we calculate $\hat{\mathbf{t}}_f$ using equations (17) and (18).

4.5. Solution Ambiguity Removal

In order to solve for A from Q, in equation (34) we took the square root of the diagonal eigenvalue matrix Λ. It is clear, however, that any matrix of the form $A\begin{bmatrix}\pm1 & 0 & 0 \\ 0 & \pm1 & 0 \\ 0 & 0 & \pm1\end{bmatrix}$ would produce the same matrix Q when multiplied by its transpose. This sign ambiguity in the first two columns of A was removed when we aligned the world coordinate axes with the first frame's camera axes, at the end of Section 4.4. However, the ambiguity in the third column of A remains. This affects the final shape simply by negating the z-component, and effects the rotation by negating the third column of the matrix M.

Geometrically, this ambiguity arises because paraperspective projection does not account for any different perspective deformation within the object itself. Thus it is not possible to distinguish the "front" of the object from the "back" of the object, as can be seen from Figure 5(a). However, in real scenarios there will be additional queues due to occlusion information as in Figure 5(b) and (c), or due to perspective distortion of the object, as in Figure 5(d), if the object is not too distant from the camera. Simple methods based on either of these phenomena should be sufficient to determine which of the two solutions given by the paraperspective factorization method is consistent with the image data.

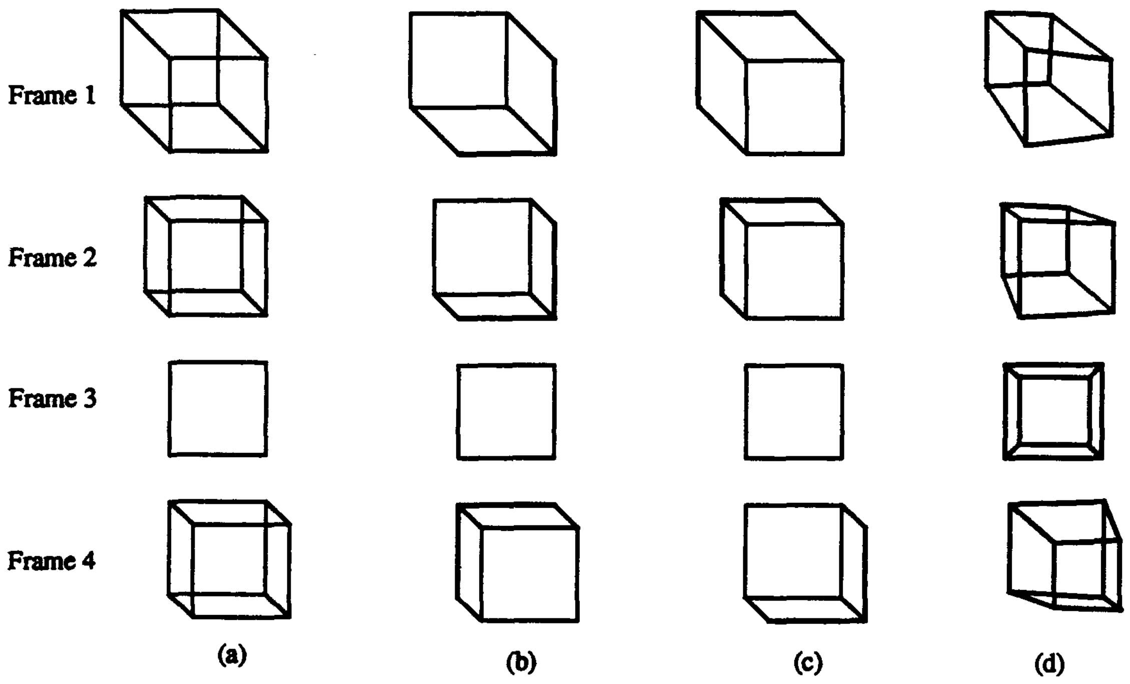

Figure 5 Ambiguity of Solution
(a) Sequence of images with two valid motion and shape interpretations.
(b), (c) Ambiguity removed due to occlusion information in the image sequence.
(d) Ambiguity removed due to perspective distortion of the object in the images

4.6. Paraperspective Projection as an Approximation to Perspective Projection

In Section 4.1., we defined paraperspective projection geometrically. We can arrive at the same equations mathematically as a first-order approximation to the perspective projection equations. The perspective projection equations are

$$x_{fp} = l\frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_{fp}} \qquad (42)$$
$$y_{fp} = l\frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_{fp}}$$

where

$$z_{fp} = \mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f) \qquad (43)$$

For simplicity we assume unit focal length, $l = 1$.

Taking the Taylor series expansion of these equations about the point

$$z_{fp} = z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \qquad (44)$$

yields

$$x_{fp} = \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(z_{fp} - z_f) + 2\frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(z_{fp} - z_f)^2 + \ldots \qquad (45)$$
$$y_{fp} = \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^2}(z_{fp} - z_f) + 2\frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f^3}(z_{fp} - z_f)^2 + \ldots$$

Ignoring all but the first term of the Taylor series yields the equations for scaled orthographic projection (See Appendix I.) However, instead of arbitrarily stopping at the first term, we can eliminate higher order terms based on the approximation that $|\mathbf{s}_p|^2/z_f^2 = 0$. Using this approximation, and combining equations (43) and (44) to determine that $z_{fp} - z_f = \mathbf{k}_f \cdot \mathbf{s}_p$, gives the following equations

$$x_{fp} = \frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{i}_f \cdot (-\mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p) \qquad (46)$$
$$y_{fp} = \frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{z_f} - \frac{\mathbf{j}_f \cdot (-\mathbf{t}_f)}{z_f^2}(\mathbf{k}_f \cdot \mathbf{s}_p)$$

Simply factoring out the $1/z_f$ and expanding the dot-products $\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)$ and $\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)$ gives

$$x_{fp} = \frac{1}{z_f}\left(\mathbf{i}_f \cdot \mathbf{s}_p + \frac{\mathbf{i}_f \cdot \mathbf{t}_f}{z_f}(\mathbf{k}_f \cdot \mathbf{s}_p) - (\mathbf{i}_f \cdot \mathbf{t}_f)\right) \qquad (47)$$
$$y_{fp} = \frac{1}{z_f}\left(\mathbf{j}_f \cdot \mathbf{s}_p + \frac{\mathbf{j}_f \cdot \mathbf{t}_f}{z_f}(\mathbf{k}_f \cdot \mathbf{s}_p) - (\mathbf{j}_f \cdot \mathbf{t}_f)\right)$$

This is equivalent to the paraperspective projection equations given by equation (15).

The approximation that $|\mathbf{s}_p|^2/z_f^2 = 0$ preserves a portion of the second term of the Taylor series expansion, the term which is of order $(|\mathbf{s}_p||\mathbf{t}_f|)/z_f^2$, while ignoring the remaining portion of the second term, which is of order $|\mathbf{s}_p|^2/z_f^2$, and all higher order terms. Clearly if the translation that the object undergoes is also small, then there is little justification for preserving this portion of the second term and not the other. In such cases, the entire second term can be safely ignored, leaving only the equations for scaled orthographic projection.

One will notice that we did not explicitly set the world origin at the object's center-of-mass as we did in Section 4.1. However, the assumption that $|\mathbf{s}_p|^2/z_f^2 = 0$ will be most accurate when the magnitudes of the $\mathbf{s}_p$ are smallest. Since the $\mathbf{s}_p$ vectors represent the vectors from the world origin to the object points, their magnitudes will be smallest when the world origin is located at the object's center-of-mass.

5. Experiments

5.1. Parameters

To test our paraperspective factorization method, we created synthetic point sequences using a perspective projection model of objects undergoing rotation and translation. We then perturbed the coordinates of each point by adding gaussian noise. We used three different object shapes; a box object whose edges were of unit length, a sphere with unit diameter, and a random point cloud within a unit cube, each containing approximately 60 feature points. All of the test runs consisted of 60 image frames of the object rotating through a total of 30 degrees each of roll, pitch, and yaw.

Our goal was not only to ascertain the performance of the paraperspective factorization method, but to determine its performance in comparison to other methods. We used five different methods to recover the shape and motion of the object and compared the accuracy of the results. The first method was the orthographic factorization method. The second was the scaled orthographic factorization method, which is described in Appendix I. The third method was our paraperspective factorization method. The fourth method was a full perspective method, which iteratively solves the perspective projection equations, as described in Appendix II. For this method, we used the paraperspective factorization method to determine an initial value. Because such methods are generally very sensitive to initial conditions, our fifth method used the same iterative solution technique, but we used the true shape and motion as initial values. This method is unfortunately not an option in real systems, but indicates what is essentially an upper bound on the accuracy achievable using a least sum-of-squares difference formulation of the full perspective projection model, without making any assumptions about the motion or object shape.

In particular, we were interested in how the methods compare as a function of the noise level in the image and the distance of the object from the camera. Thus the depth, representing the distance from the camera's focal point to the front of the object in the first frame, was varied from 3 to 60 times the object size. In generating our synthetic images, for each depth we chose the largest focal length which would keep the object in the field of view. We varied the standard deviation of the gaussian noise from 0 to 4 pixels (of a 512x512 pixel image) standard deviation. For each combination of object, depth, and noise, we performed three tests, using different random noise each time, and averaged the resulting errors.

5.2. Error Measurement

We present here the total shape error, rotation error, X-Y offset error, and Z offset (depth) error. The term "offset" refers to translations along the camera components; the X offset is $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{i}}_f$, the Y offset is $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{j}}_f$, and the Z offset is $\hat{\mathbf{t}}_f \cdot \hat{\mathbf{k}}_f$. These are the components of $\hat{\mathbf{t}}_f$ actually recovered by the methods, and although a simple transformation could recover the three elements of $\hat{\mathbf{t}}_f$ in world coordinates, the resulting translation errors would depend on the accuracy of the recovered rotation. Note that the shape and translation can only be determined up to a scaling factor (it's not possible to tell whether the image sequence is of a house fifty meters away or of a 1/10 scale model of a house five meters away). To compute the shape error, we find a scale which minimizes the root-mean-square (RMS) error between the true and computed shape, and then return this error. We use the same method for the X-Y offset, and for the Z offset. This does not correspond exactly to a valid solution, because in reality there is only a single scaling factor for all of the shape and offset values, but this avoids cases in which, for example, a large shape error influences the scaling factor, which then causes the offset error to be reported as large. The rotation error is computed as the RMS of the size in radians of the angle by which a computed camera frame must be rotated about some axis to produce the true camera frame.

5.3. Results

In the experiments in which the object was centered in the image and there was no depth translation, we found that the orthographic factorization method performed well, and the paraperspective factorization method provided no significant improvement. However, in image sequences in which there was depth translation or the object was not centered in the image, the paraperspective method performed significantly better than the orthographic factorization method. The results of these experiments are shown on the following pages. The average error results were very similar for all of the objects, so our graphs show the average over all runs of all objects. Since there were 3 objects and we performed 3 runs per object with different random noise, each data point represents the average of 9 runs.

Figure 6 shows how the various methods performed. In these image sequences, the object moved across the screen one unit horizontally and one unit vertically. The object also moved away from the camera by a total of one half the object's initial distance from the camera. Thus in a test case in which the object's depth in the first frame was 3.0, its depth in the last frame was 4.5. These experiments were done using a noise standard deviation of 2 pixels, which we consider a rather high noise level. At low depths, perspective distortion is a significant source of error in the computed results. Interestingly, our experiments show that for objects farther from the camera than 7 times the object size, refining the paraperspective solution using the perspective iteration technique improves the rotation and translation very little. However, even at depths beyond 30 times the object size, the perspective refinement method significantly improves the shape.

The behavior of the paraperspective factorization method over a range of noise levels is shown in Figure 7. We see that once the object is far enough from the camera that perspective effects are minor, the error in the computed solution is nearly proportional to the amount of noise in the input. For some reason which we are currently unable to explain, the rotation error seems to vary widely even at a given noise level.

We implemented the methods in C and performed the experiments on a Sun 4/65. Solving the system of 60 frames and 60 points required about 20-24 seconds for each of the three factorization methods, with much of this time spent computing the singular value decomposition of the measurement matrix. The perspective iteration method, however, required about 250 seconds to solve the same system.

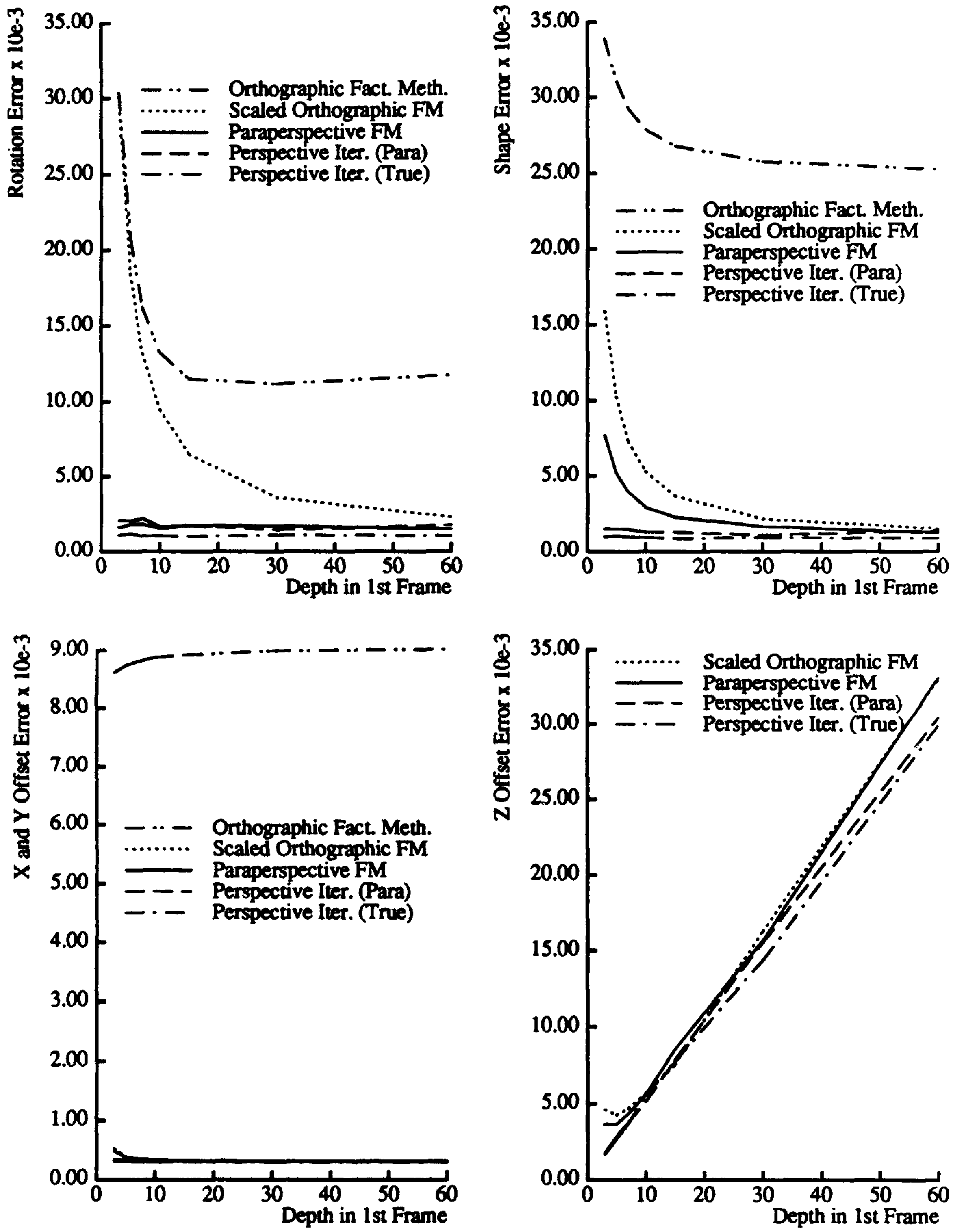

Figure 6 Comparison of Methods for a Typical Case

These figures show the error in the computed solution for the five algorithms, as a function of the object's distance from the camera. The image sequences contained translation across the image and away from the camera, and contained added gaussian noise of 2 pixels standard deviation.

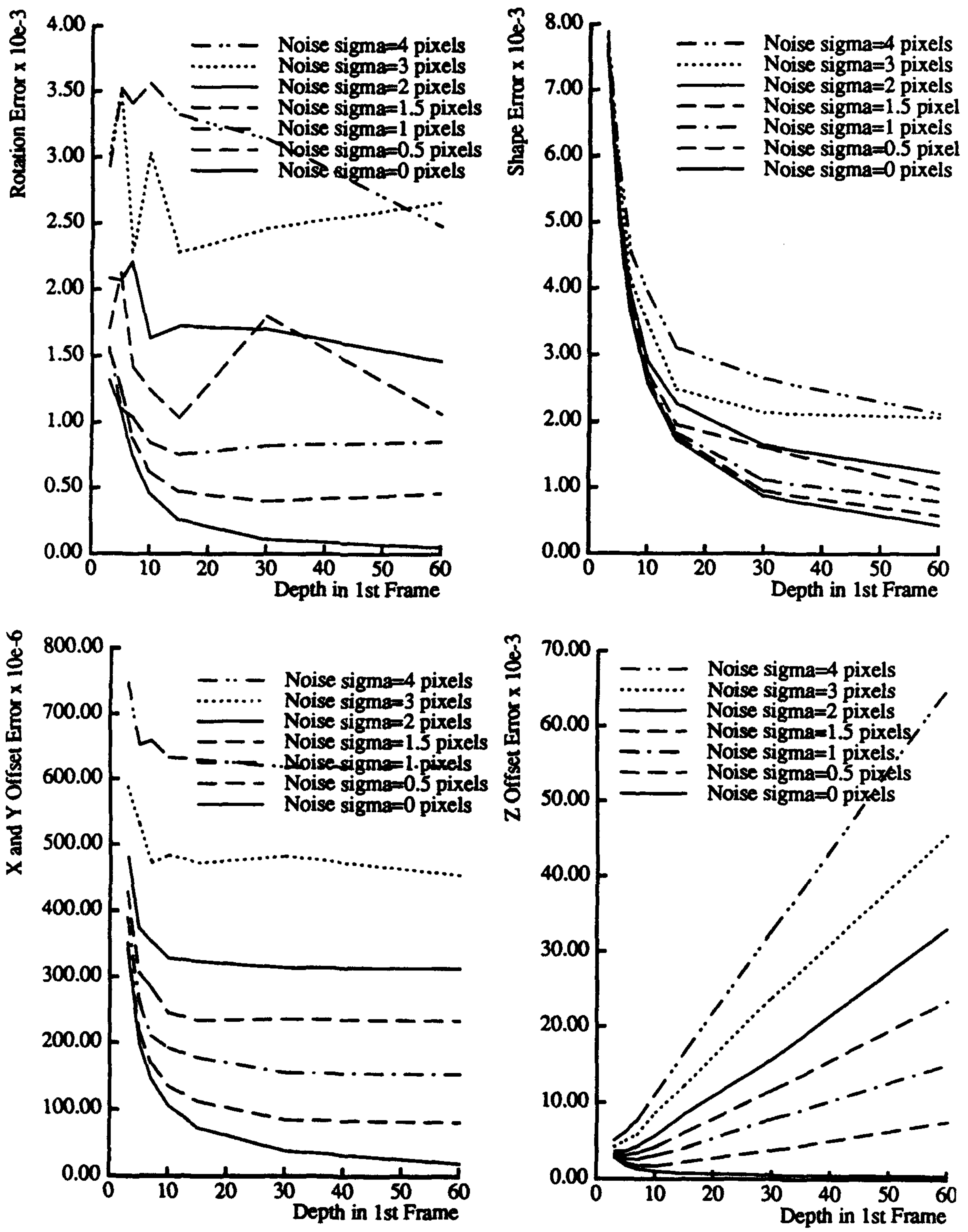

Figure 7 Paraperspective Shape and Motion Recovery by Noise Level

These figures show the error in the solution computed by the paraperspective factorization method for a range of different noise levels, as a function of the object's distance from the camera. These sequences contained translation across the image and away from the camera.

6. Conclusions

The principle that the measurement matrix has rank 3, as put forth by Tomasi and Kanade in [7], depended on the use of an orthographic projection model. We have shown in this paper that this important result also holds for the case of paraperspective projection, which closely approximates perspective projection, and have devised a paraperspective factorization method based on this projection model.

In general image sequences in which the object being viewed translates significantly toward or away from the camera or across the camera's field of view, the paraperspective method performs significantly better than the method based on orthographic projection. In image sequences in which the orthographic assumption is valid, where the object is centered in the image and not translating along the camera's optical axis, the orthographic factorization method often produces better results than the paraperspective or even the full perspective methods. This is because the orthographic method is in effect using known information about the motion in the sequence, adding the constraint that the object does not move toward or away from the camera. In images sequences in which the object is close to the camera, the paraperspective factorization method still provides accurate motion results, and provides a good approximation of the object shape; this solution can be further refined using an iterative perspective method.

The paraperspective factorization method computes the distance from the camera to the object, which enables its use in a wider range of scenarios. The method performs well in a wide range of motion scenarios, is efficient, and is robust with respect to noise.

In real image sequences, feature points will often become occluded and new feature points will appear. This problem of occlusion has been handled for the orthographic factorization method [8], but has not been addressed here for the paraperspective factorization method. We will similarly extend the paraperspective factorization method to accommodate occlusion.

So far we have concentrated on testing the method on synthetic image sequences in order to observe its performance across a very wide range of situations and noise levels, and have performed only preliminary testing on real image sequences. In order to verify the method's practicality, we plan to perform more extensive tests on real image sequences.

References

[1] John Y. Aloimonos, *?erspective Approximations*, Image and Vision Computing, 8(3):177-192, August 1990.

[2] T. Broida, S. Chandrashekhar, and R. Chellappa, *Recursive 3-D Motion Estimation from a Monocular Image Sequence*, IEEE Transactions on Aerospace and Electronic Systems, 26(4):639-656, July 1990.

[3] Joseph L. Mundy and Andrew Zisserman, *Geometric Invariance in Computer Vision*, The MIT Press, 1992, p. 512.

[4] Yu-ichi Ohta, Kiyoshi Maenobu, and Toshiyuki Sakai, *Obtaining Surface Orientation from Texels Under Perspective Projection*, Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp. 746-751, August 1981.

[5] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, 1988.

[6] Camillo Taylor, David Kriegman, and P. Anandan, *Structure and Motion From Multiple Images: A Least Squares Approach*, IEEE Workshop on Visual Motion, pp. 242-248, October 1991.

[7] Carlo Tomasi and Takeo Kanade, *Shape and Motion from Image Streams: a Factorization Method - 2. Point Features in 3D Motion*, Technical Report CMU-CS-91-105, Carnegie Mellon University, Pittsburgh, PA, January 1991.

[8] Carlo Tomasi and Takeo Kanade, *Shape and Motion from Image Streams: a Factorization Method*, Technical Report CMU-CS-91-172, Carnegie Mellon University, Pittsburgh, PA, September 1991.

[9] Roger Tsai and Thomas Huang, *Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6(1):13-27, January 1984.

Appendix I. Scaled Orthographic Factorization Method

Scaled orthographic projection, also known as "weak perspective" [3], is a closer approximation to perspective projection than orthographic projection, yet not as accurate as paraperspective projection. This method can be used when the object remains centered in the image, or when the distance to the object is large relative to the size of the object.

I.1. Scaled Orthographic Projection Equations

In the scaled orthographic projection model, object points are projected along an axis parallel to the camera's optical axis, onto a plane perpendicular to the optical axis and passing through the object's center-of-mass $\mathbf{c}$. This image is then projected onto the image plane using perspective projection, but because the points are on a plane parallel to the image plane, this is equivalent to simply scaling the points by the distance along the optical axis from the camera to the object's center-of-mass.

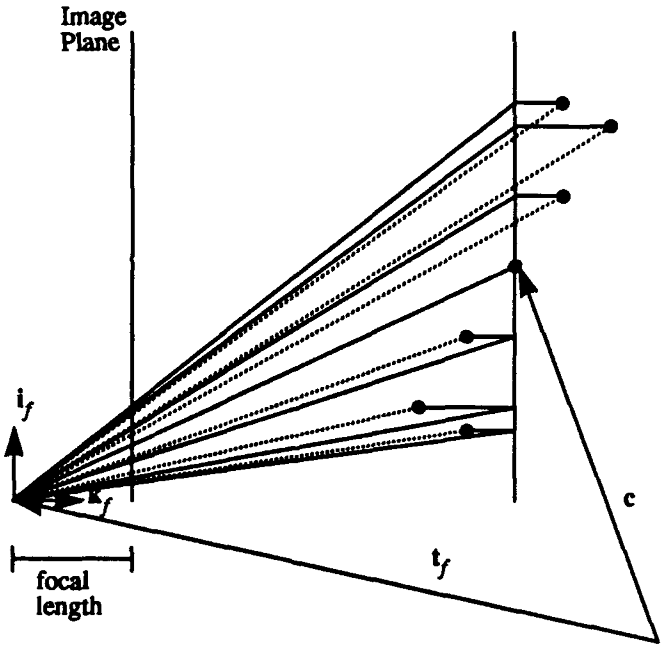

Figure 8
Scaled Orthographic Projection in two dimensions
Dotted lines indicate true perspective projection

This changes the orthographic projection equations by adding a scaling factor dependent on the distance to the object's center-of-mass, which is given by

$$z_f = (\mathbf{c} - \mathbf{t}_f) \cdot \mathbf{k}_f. \tag{48}$$

Thus the scaled orthographic projection equations are

$$x_{fp} = \frac{l}{z_f}(\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)) \qquad y_{fp} = \frac{l}{z_f}(\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)) \tag{49}$$

To simplify the equations we assume unit focal length, $l = 1$. Because the world origin is arbitrary, we set the world origin at the object's center-of-mass, so that $\mathbf{c} = 0$, and rewrite the above equations as

$$x_{fp} = \mathbf{m}_f \cdot \mathbf{s}_p + cx_f \qquad y_{fp} = \mathbf{n}_f \cdot \mathbf{s}_p + cy_f, \tag{50}$$

where

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{51}$$

$$cx_f = -\frac{\mathbf{t}_f \cdot \mathbf{i}_f}{z_f} \qquad cy_f = -\frac{\mathbf{t}_f \cdot \mathbf{j}_f}{z_f} \tag{52}$$

$$\mathbf{m}_f = \frac{\mathbf{i}_f}{z_f} \qquad \mathbf{n}_f = \frac{\mathbf{j}_f}{z_f}. \tag{53}$$

I.2. Decomposition

Because equation (50) is identical to equation (2), the measurement matrix W can still be written as $W = MS + T$ just as in orthographic case. We can still compute cx_f and cy_f immediately from the image data using equation (24), and use singular value decomposition to factor the registered measurement matrix W^* into the product of $\hat{M}$ and $\hat{S}$.

I.3. Normalization

Again, the decomposition is not unique and we must determine the 3×3 matrix A which produces the actual motion matrix $M = \hat{M}A$ and the shape matrix $S = A^{-1}\hat{S}$. We see from equation (53) that

$$|\mathbf{m}_f|^2 = \frac{1}{z_f^2} \qquad |\mathbf{n}_f|^2 = \frac{1}{z_f^2}. \tag{54}$$

We do not know the value of the depth z_f, so we cannot impose individual constraints on $\mathbf{m}_f$ and $\mathbf{n}_f$ as we did in the orthographic case. Instead, we combine the two equations as we did in the paraperspective case, to impose the following constraint:

$$|\mathbf{m}_f|^2 = |\mathbf{n}_f|^2. \tag{55}$$

Because $\mathbf{m}_f$ and $\mathbf{n}_f$ are just scalar multiples of $\mathbf{i}_f$ and $\mathbf{j}_f$, we can still use the constraint that

$$\mathbf{m}_f \cdot \mathbf{n}_f = 0. \tag{56}$$

As in the paraperspective case, equations (55) and (56) are homogeneous constraints, which could be trivially satisfied by the solution $M = 0$, so to avoid this solution we add the somewhat arbitrary consuraint that

$$|\mathbf{m}_1| = 1. \tag{57}$$

Equations (55), (56), and (57) are the scaled orthographic version of the *metric constraints*. We can compute the 3×3 matrix A which best satisfies them very easily, because the constraints are linear in the 6 unique elements of the symmetric 3×3 matrix $Q = A^T A$.

I.4. Shape and Motion Recovery

Once the matrix A has been found, the shape can easily be computed from equation (9). We compute the motion parameters by

$$\hat{\mathbf{i}}_f = \frac{\mathbf{m}_f}{|\mathbf{m}_f|} \qquad \hat{\mathbf{j}}_f = \frac{\mathbf{n}_f}{|\mathbf{n}_f|}. \tag{58}$$

Unlike the orthographic case, we can now compute the depth to the object center-of-mass in each frame, using either expression in equation (54), or by using the geometric mean of the two solutions,

$$z_f = \sqrt{\frac{1}{|\mathbf{m}_f||\mathbf{n}_f|}}. \tag{59}$$

Appendix II. Perspective Method

This section presents the iterative method used to recover the shape and motion using a perspective projection model. Although our algorithm was developed independently and handles the full three dimensional case, this method is quite similar to a two dimensional algorithm developed by Taylor, Kriegman, and Anandan as reported in [6].

II.1. Perspective Projection Equations

In the perspective projection model, sometimes referred to as the pinhole camera model, each point is projected onto the image plane directly towards the focal point of the camera. An object point's image coordinates are determined by the position at which the line connecting the object point with the camera's focal point intersects the image plane. This is illustrated in Figure 9.

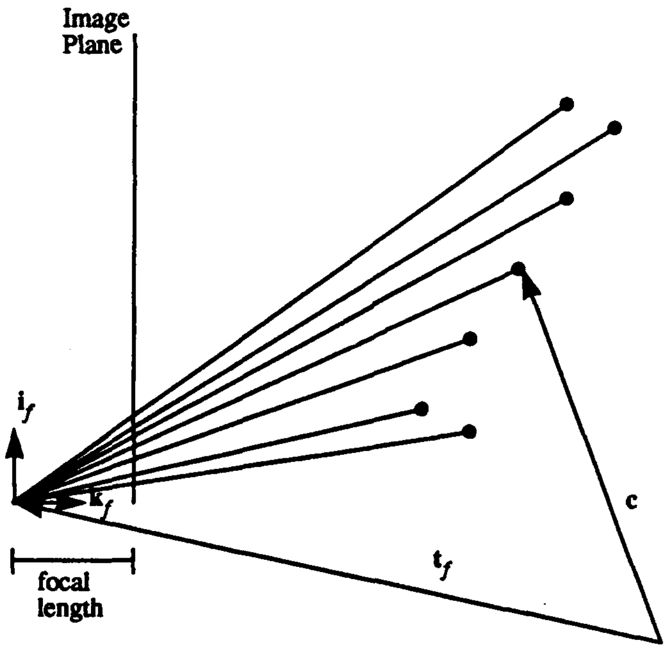

Figure 9
Perspective Projection in two dimensions

Simple geometry using similar triangles, gives us

$$x_{fp} = l\frac{\mathbf{i}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{\mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}$$
$$y_{fp} = l\frac{\mathbf{j}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)}{\mathbf{k}_f \cdot (\mathbf{s}_p - \mathbf{t}_f)} \qquad (60)$$

We now assume unit focal length and rewrite this as

$$x_{fp} = \frac{\mathbf{i}_f \cdot \mathbf{s}_p + cx_f}{\mathbf{k}_f \cdot \mathbf{s}_p + cz_f}, \qquad y_{fp} = \frac{\mathbf{j}_f \cdot \mathbf{s}_p + cy_f}{\mathbf{k}_f \cdot \mathbf{s}_p + cz_f} \tag{61}$$

where

$$cx_f = -\mathbf{i}_f \cdot \mathbf{t}_f \qquad cy_f = -\mathbf{j}_f \cdot \mathbf{t}_f \qquad cz_f = -\mathbf{k}_f \cdot \mathbf{t}_f. \tag{62}$$

II.2. Iterative Minimization Method

For this method, we take advantage of the fact that $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$ can be written as functions of only three independent rotational parameters.

$$\begin{bmatrix} \mathbf{i}_f & \mathbf{j}_f & \mathbf{k}_f \end{bmatrix} = \begin{bmatrix} \cos\alpha_f \cos\beta_f & (\cos\alpha_f \sin\beta_f \sin\gamma_f - \sin\alpha_f \cos\gamma_f) & (\cos\alpha_f \sin\beta_f \cos\gamma_f + \sin\alpha_f \sin\gamma_f) \\ \sin\alpha_f \cos\beta_f & (\sin\alpha_f \sin\beta_f \sin\gamma_f + \cos\alpha_f \cos\gamma_f) & (\sin\alpha_f \sin\beta_f \cos\gamma_f - \cos\alpha_f \sin\gamma_f) \\ -\sin\beta_f & \cos\beta_f \sin\gamma_f & \cos\beta_f \cos\gamma_f \end{bmatrix} \tag{63}$$

Thus there are six motion parameters, cx_f, cy_f, cz_f, α_f, β_f, and γ_f, for each frame, and three shape parameters, $\mathbf{s}_p = \begin{bmatrix} s_{p1} & s_{p2} & s_{p3} \end{bmatrix}$ for each point. Equation (61) gives us an overconstrained set of $2FP$ equations in these $6F + 3P$ variables. Because the perspective projection equations are non-linear, there is no apparent way to combine the equations for all points and frames into a single matrix equation, as we did in the orthographic and paraperspective case, or to separate the shape from the motion, or to compute the translational components directly from the image measurements. Instead, we formulate the problem as a general least-squares problem in which we want to determine the values of the $6F + 3P$ variables which minimize

$$\varepsilon = \sum_{f=1}^{F} \sum_{p=1}^{P} \left\{ \left(x_{fp} - \frac{\mathbf{i}_f \cdot \mathbf{s}_p + cx_f}{\mathbf{k}_f \cdot \mathbf{s}_p + cz_f} \right)^2 + \left(y_{fp} - \frac{\mathbf{j}_f \cdot \mathbf{s}_p + cy_f}{\mathbf{k}_f \cdot \mathbf{s}_p + cz_f} \right)^2 \right\}. \tag{64}$$

We could in theory apply any one of a number of non-linear equation solution techniques to this problem. Such methods begin with a set of initial starting values, and iteratively refine those values to reduce the error. Because we know the form of the equations, we can use derivative information to guide our numerical search. However, general non-linear least square techniques would not take full advantage of the structure of our equations. For every frame f and point p, we would have to compute $(\partial x_{fp}) / (\partial v)$ and $(\partial y_{fp}) / (\partial v)$ for each of the $6F + 3P$ variables v. However, in our equations most of these derivatives are zero. In fact, For a given frame f and point p, these partial derivatives are non-zero only for eight of the $6F + 3P$ variables.

Our solution technique takes advantage of the particular structure of the equations by separately refining the shape and motion parameters. We hold the shape constant and solve for the motion parameters which minimize the error. We then hold the motion constant, and solve for the shape parameters which minimize the error. We repeat this process until an iteration produces no significant reduction in the total sum of squares error. While holding the shape constant, the minimization can be done independently for each frame, which involves minimizing the error in a system of 6 variables in P equations. Likewise while holding the motion constant, we can solve for the shape separately for each point by solving a system of $2F$ equations in 3 variables. This not only reduces the problem to manageable complexity, but as pointed out in [6], it lends itself well to parallel implementation.

We perform the individual minimizations, fitting six motion variables to P equations or fitting three shape variables to $2F$ equations, using the Levenberg-Marquardt method [5], a method which uses steepest descent when far from the minimum and varies continuously towards the inverse-Hessian method as the minimum is approached.

In fact we do not need to vary all $6F+3P$ variables. Recall that the solution is only determined up to a scaling factor, the world origin is arbitrary, and the world coordinate orientation is arbitrary. These factors reduce the number of variable assignments representing different real solutions. We could choose to arbitrarily keep the first frame's rotation fix at zero roll, pitch, and yaw, and similarly fix some shape or translation parameters. However, we found in our experiments that the algorithm converged significantly faster if all of the shape and motion parameters were allowed to vary. Once the algorithm converges to a solution, we adjust the final shape and translation to place the origin at the object's center-of-mass, scale the solution so that the depth in the first frame is 1, and rotate the solution so that $\hat{i}_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ and $\hat{j}_1 = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$, or equivalently in this formulation, so that $\alpha_1 = \beta_1 = \Gamma_1 = 0$.

One problem with this sort of iteration method is that its final result can be highly dependent on its initial values. Taylor, Kriegman, and Anandan [6] require some basic odometry measurements as might be produced by a navigation system to use as initial values for their motion parameters, and use the 2D shape of the object in the first image frame, assuming constant depth, as their initial shape. To avoid the requirement for odometry measurements, which may not be available in many situations, we use the paraperspective factorization method to supply initial values to the perspective iteration method.